AFRL-IF-RS-TR-2005-214
Final Technical Report
May 2005

REDUCED-COMPLEXITY MODELS FOR NETWORK PERFORMANCE PREDICTION

University of Illinois at Urbana-Champaign

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. K144

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2005-214 has been reviewed and is approved for publication

APPROVED: /s/
KEVIN A. KWIAT
Project Engineer

FOR THE DIRECTOR: /s/
WARREN H. DEBANY, JR.
Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 2005	**Final** **May 00 – Dec 04**

4. TITLE AND SUBTITLE

REDUCED-COMPLEXITY MODELS FOR NETWORK PERFORMANCE PREDICTION

5. FUNDING NUMBERS

G - F30602-00-2-0542
PE - 62301E
PR - K144
TA - 18
WU - A1

6. AUTHOR(S)

Rayadurgam Srikant
Bruce Hajek

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Illinois at Urbana-Champaign
Department of Electrical and Computer Engineering
1308 W. Main Street
Urbana IL 61801

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington VA 22203-1714

AFRL/IFGA
525 Brooks Road
Rome NY 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2005-214

11. SUPPLEMENTARY NOTES

DARPA Program Manager: Sri Kumar/ITO/(703) 696-2234
AFRL Project Engineer: Kevin A. Kwiat/IFGA/(315) 330-1692 Kevin.Kwiat@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

The Internet consists of thousands of nodes interconnected in complex ways, with millions of users sending traffic over the network. To understand such a complex system it is necessary to develop accurate, yet simple, models to describe the performance of the network. The models have to then be used to design new algorithms that dramatically improve network performance. In this project, a variety of new models were developed to capture many phenomena in the Internet.

14. SUBJECT TERMS

Network Modeling, Internet, Fluid Model, Game Theoretic Model, Network Performance Prediction

15. NUMBER OF PAGES 29

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Abstract

The Internet consists of thousands of nodes interconnected in complex ways, with millions of users sending traffic over the network. To understand such a complex system, it is necessary to develop accurate, yet simple, models to describe the performance of the network. The models have to be then used to design new algorithms that dramatically improve network performance. In this project, we have developed a variety of models to capture many phenomena in the Internet. These include the following:

- Deterministic fluid models to describe and analyze the performance of congestion management mechanisms in the Internet.
- Stochastic models to obtain further insight into the behavior of a single node accessed by many congestion-controlled sources and to prove that the fluid models are appropriate limits of the stochastic models when the number of users is large.
- Fluid models to design new congestion-aware routing algorithms that improve the throughput of the Internet.
- Fluid models for understanding the performance of peer-to-peer networks.
- Game-theoretic models to understand incentives to deter selfish behavior in peer-to-peer resources.
- Fluid and stochastic models to design joint scheduling and flow control algorithms which lead to fair resource allocation.
- Game-theoretic models that capture the interaction between selfish users that compete for a common pool of bandwidth.

Table of Contents

List of Figures

1 Introduction

Our goal in the project was to develop models of the Internet at multiple time-scales to capture the traffic behavior and to develop models to capture the economics of providing resources to competing users. A detailed simulation model of the Internet would require a stochastic description of the packet arrival process for various types of traffic streams. However, such models would not be amenable for analysis and would be too slow for simulation purposes. Thus, a good model should be amenable to fast simulation or analysis by identifying the critical phenomenon in the time scale of interest.

It has now been established that traffic in the Internet exhibits self-similarity. A major reason for this is the distribution of file sizes is heavy tailed. This means that most files are short, but there are few files that are extremely large. It is commonly described using the Internet "80-20" rule: 80% of the Internet's traffic is due to 20% of the files. While these precise numbers are subject to continual change, the fact remains that a few files contribute to most of the traffic. Thus, the key observation is that, to improve network performance, it is important to develop models to control the behavior of the few large files. In this project, we developed a collection of models that answered fundamental questions on the behavior of the Internet and we will describe these in the following sections.

As motivation, we now present a very simple model of congestion control in the Internet and use it to study the performance of current versions of TCP. This simple model provides a powerful argument for the importance of mathematical modelling of the Internet. Congestion control is implemented in the Internet using a *window flow control* algorithm. A source's *window* is the maximum number of unacknowledged packets that the source can inject into the network at any time. For example, if the window size is 1, then the source maintains a counter which has a maximum value of 1. The counter indicates the number of packets that it can send into the network. The counter's value is initially equal to the window size. When the source sends one packet into the network, the counter is reduced by 1. Thus, the counter in this example would become zero after each packet transmission and the source cannot send any more packets into the network till the counter hits 1 again. To increment the counter, the source waits for the destination to acknowledge that it has received the packet. This is accomplished by sending a small packet called the *ack* packet, from the destination back to the source. Upon receiving the ack, the counter is incremented by 1 and thus, the source can again send one more packet. We use the term *round-trip time (RTT)* to refer to the amount of time that elapses between the instant that the source transmits a packet and the instant at which it receives the acknowledgment for the packet. The RTT consists of three components: the propagation delay of the packet through the transmission medium (which is determined by the distance between the source and destination), the queueing delay at the routers in the network and the time taken to process a packet at the routers in the network. Typically, the processing time is negligible compared to the other two components. With a window size of 1, since one packet is transmitted during every RTT, the source's data transmission rate is $1/RTT$ packets/sec.

If the window is 2, the counter's value is initially set to 2. Thus, the source can send two back-to-back packets into the network. For each transmitted packet, the counter is decremented by 1. Thus, after the first two packet transmissions, the counter is decremented to zero. When one of the packets is acknowledged and the ack reaches the source, then the source increments the counter by 1 and can send one more packet into the network. Once the new packet is transmitted, the counter is again decremented back to zero. Thus, after each ack, one packet is sent, and then, the source has to wait for the next ack

before it can send another packet. If one assumes that the processing speed of the link is very fast and that the processing times at the source and destination are negligible, then the source can transmit two packets during every RTT. Thus, the source's transmission rate is $2/RTT$ packets/sec. From the above argument, it should be clear that, if the window size is W, then the transmission rate can be approximated by W/RTT packets/sec.

If the link capacity is c and the source's window size W is such that $W/RTT < c$, then the system will be stable. In other words, all transmitted packets will be eventually processed by the link and reach the intended destination. However, in a general network, the available capacity cannot be easily determined by a source. The network is shared by many sources which share the capacities at the various links in the network. Thus, each source has to adaptively estimate the value of the window size that can be supported by the network. The most widely-used algorithm for this purpose in the Internet today is called TCP-Reno.

The TCP-Reno algorithm is quite complicated and therefore, for our modelling purposes, we consider the following simplified version of the algorithm. Assume that there is a mechanism for the receiver to indicate to the source that a packet has been lost in the network. Then, the essential features of the TCP-Reno algorithm can be summarized as below:

- Upon receipt of an ack, the source increases its current window size, denoted by $cwnd$, as follows:

$$cwnd \leftarrow cwnd + 1/cwnd.$$

- Upon being informed of a loss, the source decreases its window size by a factor of two:

$$cwnd \leftarrow cwnd/2.$$

The key feature of TCP-Reno is that it increases its window size when it does not detect congestion which is indicated by the reception of an ack, and it decreases its window size upon detecting congestion, which is indicated by the detection of a lost packet.

We now present a differential equation model that describes the TCP-Reno congestion control algorithm. Consider N TCP-Reno sources, all with the same RTT, accessing a single link. Let $W_r(t)$ denote the window size of flow r, T be its RTT, and $q(t)$ be the fraction of packets lost at the link at time t. Then, the congestion avoidance phase of TCP-Reno can be modelled as

$$\dot{W}_r = \frac{x_r(t-T)(1-q(t-T))}{W_r} - \beta x_r(t-T)q(t-T)W_r(t), \tag{1}$$

where $x_r(t) = W_r(t)/T$ is the transmission rate. The parameter β is the decrease factor and is taken to be $1/2$ although studies show that a more precise value of β when making a continuous-time approximation of TCP's behavior is closer to $2/3$. Substituting for $W_r(t)$ in terms of $x_r(t)$ gives

$$\dot{x}_r = \frac{x_r(t-T)(1-q(t))}{T^2 x_r} - \beta x_r(t-T)q(t)x_r(t). \tag{2}$$

The loss probability $q(t)$ is a function of the arrival rate at the link. Thus, let

$$q(t) = f(y(t-T)),$$

where $f(\cdot)$ is an increasing function and $y(t)$ is the total arrival rate at the link and is given by

$$y(t) = \sum_{r=1}^{N} x_r(t).$$

The equilibrium value of x_r is easily seen to be

$$\hat{x}_r = \sqrt{\frac{1-\hat{q}}{\beta\hat{q}}}\frac{1}{T}, \tag{3}$$

where $\hat{q}$ is the equilibrium loss probability. We use $\hat{}$ to denote equilibrium values. The functional form of $f(y)$ could be quite complicated in general. Among other things, it will depend upon the assumptions on the stochastic behavior of the packet arrival process at the router. To simplify the analysis, we will assume that $f(y)$ is of the following simple form:

$$f(y) = \left(\frac{y-c}{y}\right)^{+}.$$

Thus, this form of $f(y)$ can be interpreted as a fluid approximation to the loss probability: it is equal to zero if the arrival rate is less than the capacity of the link and is otherwise equal to the fraction by which the arrival rate exceeds the link capacity. Recall that the RTT T consists of two components, namely the propagation delay T_p and the queueing delay at the router. Just like the loss probability, it is difficult to precisely capture the queueing delay using a simple analytical formula. To obtain a tractable expression for the queueing delay, we recall that the TCP-Reno protocol attempts to fill up the buffer at the router and uses the resulting packet loss to obtain congestion information. Therefore, it seems reasonable to assume that the queue is full most of time. Under this assumption, our approximation to the queueing delay takes the form B/c, where B is the buffer size at the router. Thus, for all users, the RTT is given by

$$T = T_q + \frac{B}{c}.$$

To study the stability of the congestion controller given in (2), we first linearize the system around its equilibrium point. Defining $\delta x_r = x_r - \hat{x}_r$, and $\delta q = q - \hat{q}$, the linearized form of the congestion control algorithm is given by

$$\dot{\delta x_r} = \hat{x}_r\left(\frac{1-\hat{q}}{T^2\hat{x}_r^2}\delta x_r + \frac{1}{T^2\hat{x}_r^2}\delta q + \beta\hat{q}\delta x_r + \beta\hat{x}_r\delta q\right),$$

and

$$\delta q = \frac{c}{\hat{y}^2}\sum_r \delta x_r(t-T).$$

Defining $\delta y = y - \hat{y}$, and using the equilibrium relationship (3) yields

$$\dot{\delta x_r} + \alpha_1\delta x_r + \alpha_2\delta x_r(t-T) = 0, \tag{4}$$

where

$$\alpha_1 = 2\beta\hat{q}\hat{x}_r, \qquad \alpha_2 = \beta\hat{x}_r.$$

A well-known result called Hayes' lemma states that the linearized delay-differential equation describing TCP-Reno's dynamics is stable if one of the following conditions is satisfied:

- $\alpha_1 \geq \alpha_2$,
- $\alpha_1 < \alpha_2$ and

$$\alpha_2 T \sqrt{1 - \frac{\alpha_1^2}{\alpha_2^2}} < \arccos\left(-\frac{\alpha_1}{\alpha_2}\right).$$

For the first condition to be satisfied, we require $\hat{q} \geq 1/2$. This is not a practical scenario since it requires at least half the packets to be dropped at the router. The second condition can be written as

$$\frac{c}{N} T < \frac{1}{\beta} \frac{(1-\hat{q}) \arccos(-2\hat{q}_r)}{\sqrt{1-4\hat{q}^2}}. \tag{5}$$

Note that the equilibrium relationship (3) can be rewritten as

$$\frac{(1-\hat{q})^3}{\hat{q}} = \left(\frac{c}{N} T\right)^2.$$

If we let c/N (which is simply the capacity per user) be a constant and increase the RTT, then it is clear from the previous equation that $\hat{q}$ must decrease. Thus, for large T, the right-hand side of the stability condition can be approximated by letting $\hat{q} = 0$ which gives the following condition for stability

$$\frac{c}{N} T < \frac{\pi}{2\beta}.$$

Clearly, this condition will be violated as T increases or c/N increases. From the above analysis, we can conclude that TCP-Reno is not a scalable protocol, i.e., its stability is compromised if either the RTT of the users is large or if the available capacity per user at the router is large. In the following sections, we present our results on the the use of many other such models to analyze and improve the performance of resource allocation protocols for the Internet and wireless networks. Papers resulting from the work carried out in this project can be downloaded from the following websites: http://www.comm.csl.uiuc.edu/~srikant and http://www.comm.csl.uiuc.edu/~hajek.

2 Adaptive Virtual Queue: An Active Queue Management Scheme for Internet Routers

In the modern day Internet, there has been a strong demand for QoS (Quality-of-Service) and fairness among flows. As a result, in addition to the sources, the links also play an active role in congestion control and avoidance. Random Early Discard (RED) was originally proposed to achieve fairness among sources with different burstiness and to control queue lengths. RED allows for dropping packets at a router before buffer overflow occurs. Another form of congestion notification that has been discussed since the advent of RED is Explicit Congestion Notification (ECN). ECN has been proposed to allow each link to participate in congestion control by notifying users when it detects an onset of congestion. Upon detecting incipient congestion, a bit in the packet header is set to one for the purpose of notifying the user that a link on its route is experiencing congestion. The user then reacts to the *mark* as if a

packet has been lost. Thus, the link avoids dropping the packet (thereby enhancing good throughput) and still manages to convey congestion information to the user.

To provide ECN marks or drop packets in order to control queue lengths or provide fairness, the routers have to select packets to be marked in a manner that conveys information about the current state of the network to the users. Algorithms that the routers employ to convey such information are called *Active Queue Management (AQM)* schemes. An AQM scheme might mark or drop packets depending on the policy at the router. Here, we use the term "marking" more generally to refer to any action taken by the router to notify the user of incipient congestion. The action can, in reality, be ECN-type marking or dropping (as in RED) depending upon the policy set for the router. As in earlier work on studying AQM schemes, this distinction is blurred in the mathematical analysis to allow for the development of simple design rules for the choice of AQM parameters. However, our simulations considered marking and dropping schemes separately.

Designing robust AQM schemes has been a very active research area in the Internet community. Some AQM schemes that have been proposed include RED, SRED, BLUE, Proportional Integral (PI) controller, and REM. While most of the proposed AQM schemes detect congestion based on the queue lengths at the link (e.g., RED), some AQM schemes detect congestion based on the arrival rate of the packets at the link (e.g., virtual queue-based schemes) and some use a combination of both (e.g., PI). Also, most of the AQM schemes involve adapting the marking probability (as noted before we use the term *marking* to refer to both *marking* and *dropping*) in some way or the other. An important question is how fast should one adapt while maintaining the stability of the system? Here the system refers jointly to the TCP congestion controllers operating at the edges of the network and the AQM schemes operating in the interior of the network. Adapting too fast might make the system respond quickly to changing network conditions, but it might lead to large oscillatory behavior or in the worst-case even instability. Adapting too slowly might lead to sluggish behavior and more losses or marks than desired, which might lead to a lower throughput.

In this project, we developed a virtual-queue based AQM scheme, namely the Adaptive Virtual Queue (AVQ). The motivation behind the AVQ algorithm is to design an AQM scheme that results in a low-loss, low-delay and high utilization operation at the link. We then developed a methodology for finding the fastest rate at which the marking probability adaptation can take place, given certain system parameters like the maximum delay and the number of users, so that the system remains stable. We note that the marking probability in AVQ is implicit, no marking probability is explicitly calculated and thus, no random number generation is required. On the other hand, we replace the marking probability calculation with the computation of the capacity of a virtual queue.

The AVQ algorithm maintains a virtual queue whose capacity (called *virtual capacity*) is less than the actual capacity of the link. When a packet arrives in the real queue, the virtual queue is also updated to reflect the new arrival. Packets in the real queue are marked/dropped when the virtual buffer overflows. The virtual capacity at each link is then adapted to ensure that the total flow entering each link achieves a desired utilization of the link. An appealing feature of the AVQ scheme is that, in the absence of feedback delays, the system is fair in the sense that it maximizes the sum of utilities of all the users in the network. Combining this with the fact that a TCP user r with an RTT of d_r can be approximated by a user with a utility function $\frac{-1}{d_r^2 x_r}$, where x_r is the rate of the TCP user, shows that the network as a whole converges to an operating point that minimizes $\sum_r \frac{-1}{d_r^2 x_r}$.

The criterion we use to choose the parameters is local stability of the congestion-controllers and the

AQM scheme together. We were able to show through simulations that the AVQ controller out performs a number of other well-known AQM schemes in terms of losses, utilization and average queue length. In particular, we showed that AVQ is able to maintain a small average queue length at high utilizations with minimal loss at the routers. This conclusion also holds in the presence of short flows arriving and departing at the link. We also showed that AVQ responds to changing network conditions better than other AQM schemes (in terms of average queue length, utilization and losses).

We also studied the performance of AVQ when dropping (instead of marking) is employed at the routers. While AVQ performs better than other AQM schemes in terms of utilization and average queue length, the fairness of AVQ can be improved using a probabilistic AQM scheme (like RED) on AVQ. We note that a probabilistic AQM scheme on the virtual queue is required only when the link drops packets and not when the link marks packets because multiple marks within a single window does not cause TCP to time-out or go into slow-start.

An important feature of the AVQ algorithm is that one can employ any AQM algorithm in the virtual queue. Thus, if there are desirable properties in any other marking schemes, one can easily incorporate it into the AVQ scheme. However, when marking is employed, our experience has been that a simple mark-tail would suffice.

3 Scalable Congestion Control and AQM Schemes for Arbitrary Network Topologies

Recently, there has been a flurry of research activity on decentralized end-to-end network congestion control algorithms. A widely-used framework is to associate a utility function with each flow and maximize the aggregate system utility function subject to link capacity constraints. Congestion control schemes can be viewed as decentralized source and router algorithms to drive the system operating point to the optimum or some suboptimum solution of this maximization problem.

Congestion control schemes can be divided into three classes: primal algorithms, dual algorithms and primal-dual algorithms. In primal algorithms, the users adapt the source rates dynamically based on the route prices, and the links select a static law to determine the link prices directly from the arrival rates at the links. In dual algorithms, on the other hand, the links adapt the link prices dynamically based on the link rates, and the users select a static law to determine the source rates directly from the route prices and the source parameters. Primal-dual algorithms combine these two schemes and dynamically compute both user rates and link prices.

A modified primal algorithm, called the AVQ (Active Virtual Queue) algorithm, was introduced in the previous section. Here the link prices in the original primal algorithm are slowly adjusted so that asymptotically in time, the link prices become equal to the Lagrange multipliers. More importantly, in the presence of feedback delays, the parameters of this algorithm can be chosen such that the network is locally stable. The main benefit of this algorithm is that it achieves arbitrary fairness among the users and leads to full link utilization. This idea was adopted by others to modify the dual algorithm to allow slow adaptation at the sources and achieve the same benefits as the AVQ algorithm.

Both the modified primal algorithm and the modified dual algorithm have dynamic adaptations at both sources and routers, and thus can be regarded primal-dual. However, all the algorithms in the primal family relate the network congestion measure directly with the link aggregate rate, which

corresponds to averaging the feedback from the network at the sources; and all the algorithms in the dual family relate the source rate directly with the route congestion measure, which corresponds to averaging the source rates at the links before the feedback of more explicit congestion information to the sources.

In this section, we briefly describe our work on generalizing the class of **primal-dual** algorithms, and provide design guidelines to stabilize these algorithms in general topology networks with heterogeneous feedback delays. In this class of algorithms, the source dynamics are similar to those in the primal algorithm in while the link dynamics are similar to those in the dual algorithm. We obtained a local stability result which subsumes the dual algorithm as its limiting case when the source adaptation speed approaches infinity.

From the stability analysis of the general primal-dual algorithm, we also showed that RED (Random Early Detection) could stabilize TCP-Reno if modified slightly. Our modification to RED sets the packet marking probability to be an exponential function of the length of a virtual queue whose capacity is slightly smaller than the link capacity. Due to the exponential marking profile, we call it Exponential-RED (E-RED). From our analysis, it can be shown that E-RED stabilizes TCP-Reno and all its packet loss/mark based variations. Compared with other queue-length-based AQM schemes, like RED, REM, PI and BLUE, E-RED is the first such scheme that can be proved to stabilize TCP-Reno for a general topology network with heterogeneous delays.

We performed *ns*-2 and Matlab simulations to compare E-RED with RED and to discuss the dependence of E-RED's performance on the network scenario and the E-RED parameter choices. The simulation results showed that E-RED outperforms RED when combined with TCP-Reno in the sense that it achieves less queue length oscillation, higher bandwidth utilization, and lower queueing delay at the same time. The simulation results also show that E-RED works well with other proposals for TCP in future high bandwidth networks, namely HighSpeed TCP and Scalable TCP.

4 Validity of Fluid Models of Congestion Control

As mentioned previously, deterministic fluid-flow models have been widely used to describe congestion control and active queue management (AQM) schemes in the Internet. These models capture the mean behavior of the congestion controlled sources. All of these models use a packet marking (or packet dropping) function to describe the fraction of packets marked (or dropped) at a link. Depending upon the model, the marking function is either a function of the queue length or a function of the instantaneous arrival rate at the router. In this project, we considered AQM schemes where the router decides the fraction of packets to be marked based on the occupancy level of a real or virtual queue.

We start with a stochastic model of a single link accessed by many congestion-controlled flows. Randomness in the congestion-controlled Internet may be due to many reasons:

- unresponsive flows which do not respond to congestion indication,
- the probabilistic nature of packet marking by an AQM scheme,
- asynchronous updates among sources,
- the inability to precisely model window flow control mechanism, and

- the initial ramp-up phase (for example, *slow start* in TCP flow control) of the congestion control mechanism.

In addition to deriving a deterministic model from this stochastic system under a limiting regime where the number of sources is large, we also derived a stochastic model to capture the deviations from the deterministic limit. We used the stochastic model to further study the performance of rate-based and queue-based models of AQM schemes.

Our main contributions were as follows: depending upon the manner in which a parameter in REM is scaled with the number of flows, we showed that the limiting deterministic/stochastic model of the congestion-controlled link would capture the AQM behavior using either a rate-based or a jointly rate-and-queue-based marking function. The choice of the appropriate model for the marking function is critical in designing the parameters of the congestion control/AQM scheme.

To demonstrate our results, we simulate a single bottleneck link accessed by multiple TCP sources, all of which are in the congestion avoidance phase. Apart from the TCP sources we also consider unresponsive flows. We use an ON-OFF model for the uncontrolled flows. The uncontrolled flows toggle between ON and OFF state which are exponentially distributed with mean 0.2 s. In the ON state, an uncontrolled flow sends data at a rate ρ packets/s. In all our simulations with various AQM schemes, we change N, the number of TCP sources, which is also the number of uncontrolled flows in the system. The link capacity in all our simulations is Nc, where $c = 80$ packets/s. The flow rate ρ of the uncontrolled flows in the ON state is adjusted so that uncontrolled flows deliver a load of 25% into the link. Every simulation result is averaged over 10 runs. We report simulation results with four sets of parameters as follows:

1. $\theta = 0.85$, $\gamma^{(N)} = 0.0075/N$
2. $\theta = 0.85$, $\gamma^{(N)} = 0.0075$,
3. $\theta = 1$, $\gamma^{(N)} = 0.05/N$,
4. $\theta = 1$, $\gamma^{(N)} = 0.05$,

where θ is the fraction of the virtual queue capacity as a function of the link capacity, and $\gamma^{(N)}$ is the REM parameter, i.e., when the queue length is q, the marking probability is $1 - e^{-\gamma^{(N)} q}$, and N is the number of TCP users.

We first show results for the case $\theta = 0.85$, i.e., when the capacity of the virtual queue is $0.85Nc$, N being the number of TCP flows in the system. We compare the average throughput obtained from the simulation with the predicted equilibrium of the suitable limiting model for two different parameter scalings of REM: $\gamma^{(N)} = 0.0075/N$ and $\gamma^{(N)} = 0.0075$. The plots are shown in Figure 1. The equilibrium point of the suitable limiting models predict the average throughput into the link reasonably accurately. Further, if the capacity of the virtual queue is 0.85 fraction of the link capacity, it is possible to attain a mean queue-length (at the real-queue) that does not grow with N, and thus, providing a queueing delay of $O(1/N)$. Such a behavior can be observed in the both the regimes of $\gamma^{(N)}$ considered in the plots.

Next we consider the case $\theta = 1$. Observe that $\theta = 1$ is equivalent to marking packets based on the occupancy of the real-queue. In Figure 2, we show the plots of average throughput at the link and

the mean queue-length for $\theta = 1$ with two different parameters scalings of $\gamma^{(N)}$: $\gamma^{(N)} = 0.05/N$ and $\gamma^{(N)} = 0.05$. Note that, in this case, an $O(1)$ queue length (and thus a queueing delay of $O(1/N)$) at the real-queue is obtained only in the parameter regime $\gamma^{(N)} = 0.05$.

Based on the two sets of plots, we summarize our observations as follows:

- If the capacity of the virtual queue is less than that of the link capacity, it is possible to attain a negligible queueing delay in the either parameters regimes of $\gamma^{(N)}$. The limiting models as obtained in the previous sections quite accurately predict the equilibrium values.

- If the capacity of the virtual queue is identical to the link capacity, simulations suggests that negligible queueing delay can be obtained only in the parameter regime $\gamma^{(N)} = \gamma$. In this case, the appropriate limiting model is a rate-based marking model even though marking may be implemented based on the contents of the queue.

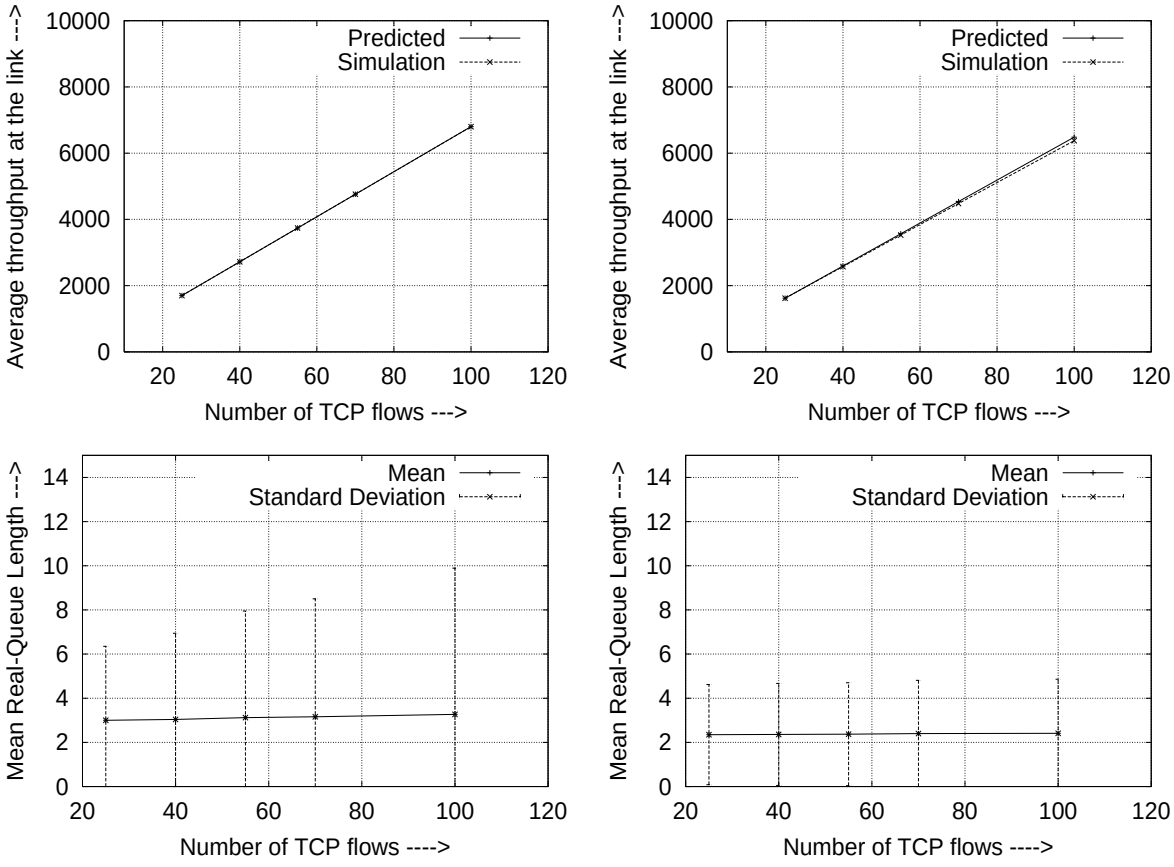

Figure 1: Comparison of average utilization, coefficient of variation and mean queue length with virtual queue based REM. On the left-hand panel we show plots when $\gamma^{(N)}$ is scaled as $\gamma^{(N)} = 0.0075/N$, and the right-hand panel shows plots with $\gamma^{(N)} = 0.0075$.

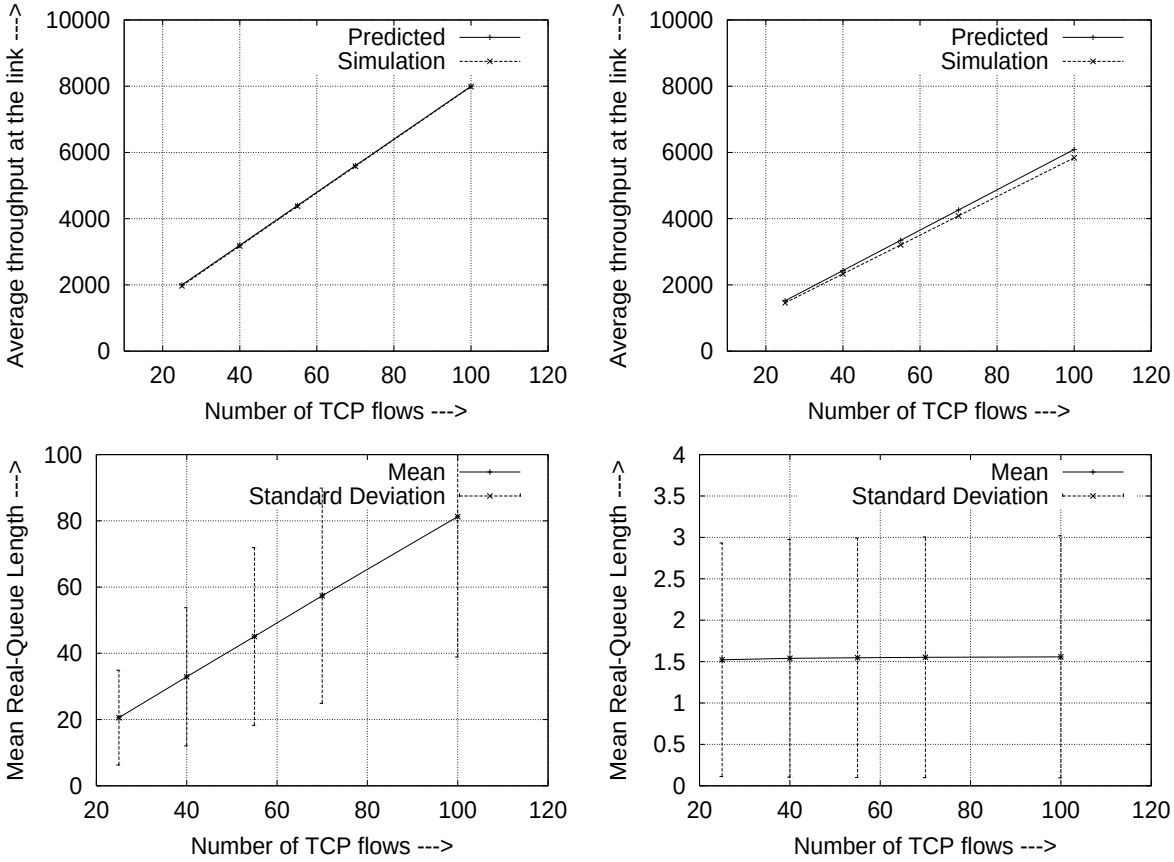

Figure 2: Comparison of average throughput, and mean queue length with virtual queue based REM with $\theta = 1$. On the left-hand panel we show plots when $\gamma^{(N)}$ is scaled as $\gamma^{(N)} = 0.05/N$, and the right-hand panel shows plots with $\gamma^{(N)} = 0.05$.

5 Modeling Peer-to-Peer Networks

Peer-to-Peer (P2P) applications have become immensely popular in the Internet. Traffic measurements shows that P2P traffic is starting to dominate the bandwidth in certain segments of the Internet. Among P2P applications, file sharing is perhaps the most popular application. Compared to traditional client/sever file sharing (such as FTP, WWW), P2P file sharing has one big advantage, namely, scalability. The performance of traditional file sharing applications deteriorates rapidly as the number of clients increases, while in a well-designed P2P file sharing system, more peers generally means better performance. There are many P2P file sharing programs, such as Kazza, Gnuttella, eDonkey/overnet, BitTorrent, to name a few. In this project, we developed simple models to understand and study the behavior of BitTorrent which is proving to be one of the more popular P2P applications today.

For a BitTorrent network (or a general P2P file sharing network), several issues have to be addressed in order to understand the behavior of the system.

- *Peer Evolution:* In P2P file sharing, the number of peers in the system is an important factor in determining network performance. Therefore, it is useful to study how the number of peers evolves as a function of the request arrival rate, the peer departure rate, the uploading/downloading bandwidth of each peer, etc.

- *Scalability:* To realize the advantages of P2P file sharing, it is important for the network performance to not deteriorate, and preferably to actually improve, as the size of the network increases. Network performance can be measured by the average file downloading time and the size of the network can be characterized by the number of peers, the arrival rate of peers, etc.

- *File Sharing Efficiency:* It is common for peers in a P2P network to have different uploading/downloading bandwidths. Further, in BitTorrent-like systems, a file may be broken into smaller pieces and the pieces may be distributed at random among the peers in the network. To efficiently download the file, it is important to design the file-sharing protocol such that each peer is matched with others who have the pieces of the file that it needs and further, to ensure that the downloading bandwidth of each peer is fully utilized.

- *Incentives to prevent free-riding:* Free-riding is a major cause for concern in P2P networks. Free-riders are peers who try to download from others while not contributing to the network, i.e., by not uploading to others. Thus, most P2P networks try to build in some incentives to deter peers from free-riding. Once the incentive mechanism is introduced into the network, each peer may try to maximize its own net benefit within the constraints of the incentive mechanism. Thus, it is important to study the effect of such behavior on the network performance.

The basic idea of P2P network is to have peers participate in an application level overlay network and operate as both servers and clients. Since the service burden is distributed to all participating peers, the system is expected to scale well even when the network is very large. Besides file sharing, P2P overlays have also been deployed in distributed directory service, web cache, storage, and grid computation.

Our work differs from prior work in the following respects:

- Instead of developing and numerically studying a detailed stochastic model, we develop a simple deterministic model which allows us to obtain simple expressions for the average file-transfer time, thus providing insight into the performance of the P2P network. We also incorporate realistic scenarios in our fluid model such as the abandonment of file transfers by peers and download bandwidth constraints.

- Then, we develop a simple stochastic fluid model which characterizes the variability of the number of peers around the equilibrium values predicted by the deterministic fluid model.

- We also develop a simple model to study the efficiency of downloading from other peers and argue that the file-sharing protocol in BiTorrent is very efficient.

- Finally, we consider the mechanisms built into BitTorrent to avoid free-riding and analyze the impact of these mechanisms on the users' behaviors and network performance.

We now briefly describe BitTorrent. BitTorrent is a P2P application whose goal is to facilitate fast downloads of popular files. Here we provide a brief description of how BitTorrent operates when a single file is downloaded by many users. Typically the number of simultaneous downloaders for popular files could be of the order of a few hundreds while the total number of downloaders during the lifetime of a file could be of the order of several tens or sometimes even hundreds of thousands. The basic idea in BitTorrent is to divide a single large file (typically a few 100 MBytes long) into pieces of size 256 KB each. The set of peers attempting to download the file do so by connecting to several other peers simultaneously and download different pieces of the file from different peers.

To facilitate this process, BitTorrent uses a centralized software called the *tracker*. In a BitTorrent network, a peer that wants to download a file first connects to the tracker of the file. The tracker then returns a random list of peers that have the file. The downloader then establishes a connection to these other peers and finds out what pieces reside in each of the other peers. A downloader then requests pieces which it does not have from all the peers to which it is connected. But each peer is allowed to upload only to a fixed number (default is four) at a given time. Uploading is called *unchoking* in BitTorrent. Which peers to unchoke is determined by the current downloading rate from these peers, i.e., each peer uploads to the four peers that provide it with the best downloading rate even though it may have received requests from more than four downloaders. This mechanism is intended to deter free-riding. Since a peer is only uploading four other peers at any time, it is possible that a peer, say Peer A, may not be uploading to a peer, say Peer B, which could provide a higher downloading rate than any of the peers to which Peer A is currently uploading. Therefore, to allow each peer to explore the downloading rates of other peers, BitTorrent uses a process called *optimistic unchoking*. Under optimistic unchoking, each peer randomly selects a fifth peer from which it has received a downloading request and uploads to this peer. Thus, including optimist unchoking, a peer may be uploading to five other peers at any time. Optimistic unchoking is attempted once every 30 seconds and to allow optimistic unchoking while keeping the maximum number of uploads equal to five, an upload to the peer with the least downloading rate is dropped.

BitTorrent distinguishes between two types of peers, namely *downloaders* and *seeds*. Downloaders are peers who only have a part (or none) of the file while seeds are peers who have all the pieces of the file but stay in the system to allow other peers to download from them. Thus, seeds only perform uploading while downloaders download pieces that they do not have and upload pieces that they have.

Ideally, one would like an incentive mechanism to encourage seeds to stay in the system. However, BitTorrent currently does not have such a feature. We simply analyze the performance of BitTorrent as is.

In practice, a BitTorrent network is a very complicated system. There may be hundreds of peers in the system. Each peer may have different parts of the file. Each peer may also have different uploading/downloading bandwidth. Further, each peer only has partial information of the whole network and can only make decisions based on local information. In addition, BitTorrent has a protocol (called the *rarest-first policy*) to ensure a uniform distribution of pieces among the peers and protocols (called the *endgame mode*) to prevent users who have all but a few of the pieces from waiting too long to finish their download. As with any good modelling exercise, we tradeoff between the simplicity of the model and its ability to capture all facets of the protocol. Thus, we use a simple fluid model to study the scalability and the stability of the system. We then abstracted the built-in incentive mechanism of BitTorrent and studied its effect on network performance. Under certain conditions, we proved that a Nash equilibrium exists, under which each peer chooses its physical uploading bandwidth to be equal to the actual uploading bandwidth. We also briefly discussed the effect of optimistic unchoking on free-riding. Our experimental results show that the simple fluid model can capture the behavior of the system even when the arrival rate is small.

We performed a series of experiments to validate the fluid model. In the first two experiments, we compare a simulated BitTorrent-like network and the fluid model. In the last experiment, we actually introduced a seed into the BitTorrent network, studied the evolution of the seeds/downloaders, and compared it to our fluid model results. Due to copyright reasons, we obviously could not introduce a very popular file into the network. However, as we will show in our experimental results, even for a file which had a total of less than 100 completed downloads, the match between the fluid model and the observed data is quite close.

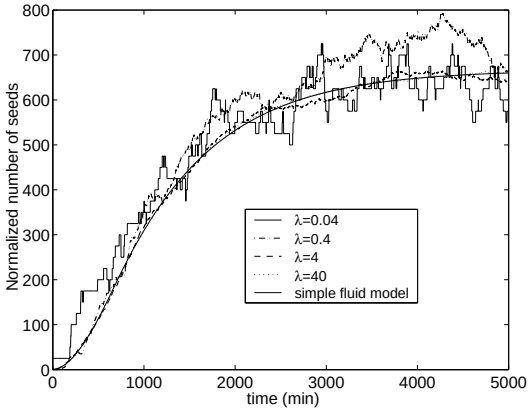

Figure 3: Experiment 1 : The evolution of the number of seeds as a function of time

Experiment 1 In Figs 3 and 4, we compare the simple deterministic fluid model that we derived with the results from a discrete-event simulation of a BitTorrent-like network. In the discrete-event simulation, we use a Markov model. We chose the following parameters for this simulation: the upload

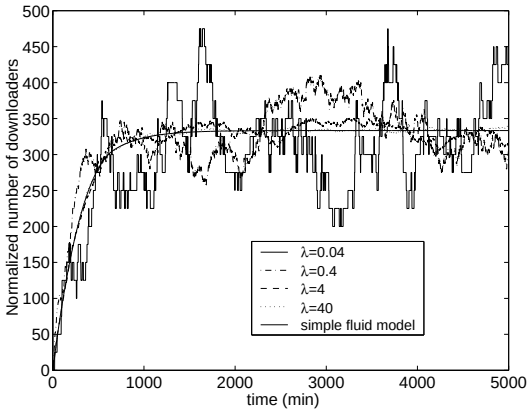

Figure 4: Experiment 1 : The evolution of the number of downloaders as a function of time

rate of a peer $\mu = 0.00125$, the download rate of a peer $c = 0.002$, the rate at which a user aborts a download θ and the rate at which a seed leaves the system γ are chosen to be $\theta = \gamma = 0.001$. When the number of downloaders is 1, we set the probability with which a contacted peer is useful to another peer, denoted by the parameter η, is taken to be zero; otherwise, we set $\eta = 1$. Initially, there is one seed and no downloader. We also keep the number of seeds no less than one during the entire simulation. We change the arrival rate λ from 0.04 to 40 and plot number of seeds/downloaders normalized by the arrival rate, i.e., $\frac{y(t)}{\lambda}$ and $\frac{x(t)}{\lambda}$, from both simulations and the fluid model. From the figures, we see that the simple fluid model is a good approximation of the system when λ is large, but the match is quite good even for small λ. The figures also indicate that the number of downloaders increases linearly with the arrival rate λ. By Little's law, this implies that the average download time is constant, independent of the peer arrival rate, which shows that the system scales very well. In other words, even very popular files can be downloaded at the same speed as less popular files.

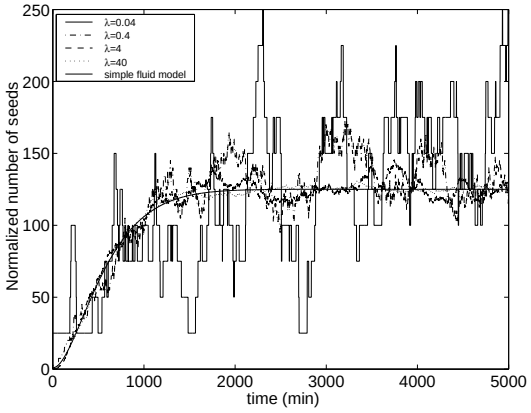

Figure 5: Experiment 2 : The evolution of the number of seeds

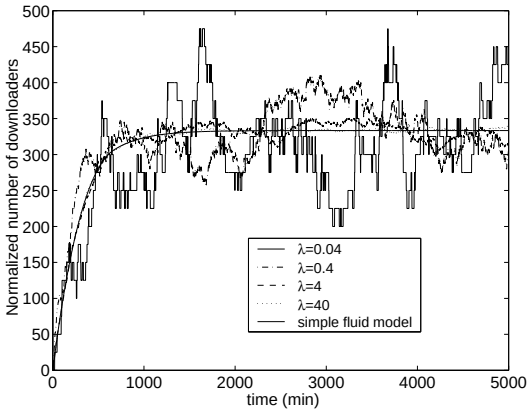

Figure 6: Experiment 2 : The evolution of the number of downloaders

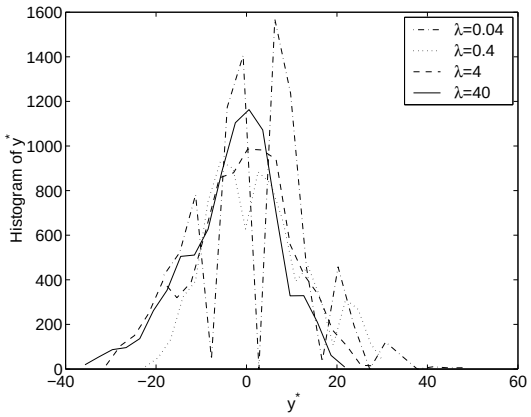

Figure 7: Experiment 2 : Histogram of the variation of the number of seeds around the fluid model

Experiment 2 In Figs. 5 and 6, we have the same setting as the first experiment, except that now we set $\gamma = 0.005$. With the change of γ, the uploading bandwidth now becomes the bottleneck. In this setting, we have the similar result as before. Again, we see that the simple fluid model is accurate when λ is large, but performs well even for smaller λ. We also plot the histogram of $\hat{x}$ and $\hat{y}$ (the deviation of number of downloaders and seeds, respectively, from their fluid model values) in Figs. 7 and 8,

$$\hat{x}(t) = \frac{x_{sim}(t) - x(t)}{\sqrt{\lambda}}$$

and

$$\hat{y}(t) = \frac{y_{sim}(t) - y(t)}{\sqrt{\lambda}},$$

where $x_{sim}(t)$ and $y_{sim}(t)$ are the number of downloaders and seeds respectively in the actual simulation and $x(t)$ and $y(t)$ are the number of downloaders and seeds in deterministic fluid model. From the theory that we developed, we expect the histograms to look roughly Gaussian and this fact is borne out

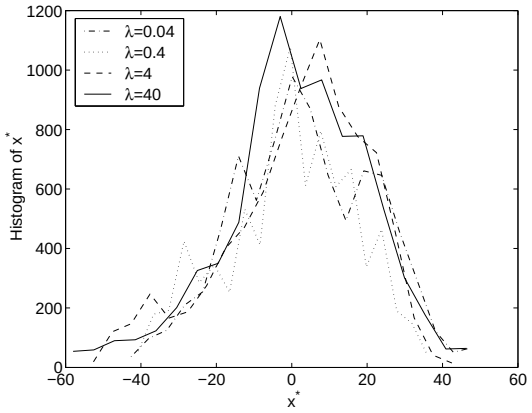

Figure 8: Experiment 2 : Histogram of the variation of the number of downloaders around the fluid model

by the figures for sufficiently large λ. We can see that the variance of $\hat{x}$ and $\hat{y}$ do not change much when λ changes from 0.04 to 40.

Experiment 3 In this experiment, we introduced a file into the BitTorrent network and collected the log files of the BitTorrent tracker for a time period of around three days. When a peer joins/leaves the system or completes the download, it reports the event to the tracker. In addition, peers regularly report information such as the total amount of data uploaded/downloaded so far, the number of bytes that still need to be downloaded, etc. The tracker keeps all the information in the log files. Hence, we can analyze the tracker log files and retrieve useful information. The parameters $\lambda, \theta,$ and γ can be measured by counting the peer arrival, the downloader departure, and the seed departure respectively. However, from the tracker log files, we cannot determine whether the uploading bandwidth or the downloading bandwidth is the bottleneck. So we assume the uploading bandwidth is the bottleneck and estimate μ by dividing the measured total uploading rate by the number of peers (i.e., we assume that $\eta = 1$). The size of the file that was introduced was around $530MB$. The average uploading bandwidth was estimated to be $90kb/s$. We use $1\ min$ as the time unit to calculate arrival rates, departure rates, etc. The normalized uploading bandwidth (normalized by the file size in bytes) was estimated $\mu = 0.0013$. The downloader leaving rate was estimated to be $\theta = 0.001$. An interesting feature that we observed in the real BitTorrent is that λ and γ are in fact time-varying. We attribute this to the fact that when a new file is introduced into the system, the first few seeds stay in the system long enough to ensure that there is a sufficient population of peers to sustain the system. If the initial seeds depart too quickly, the system will simply die, i.e., there will be no one to download from.

From the tracker logs, we estimate that, for $t \leq 800min$, $\lambda = 0.06$ and $\gamma = 0.001$. When $t \geq 1300min$, $\lambda = 0.03$ and $\gamma = 0.0044$. In between, the arrival rate increases roughly linearly. In our fluid model simulation, for time between $800min$ and $1300min$, we let λ and γ change linearly. We also set the downloading bandwidth $c = 1$ for the fluid model simulation (note that the actual value of c will not affect the fluid model results if it is above a certain threshold).

The simulation results are shown in Figs 9 and 10. The real trace is measured from the tracker log

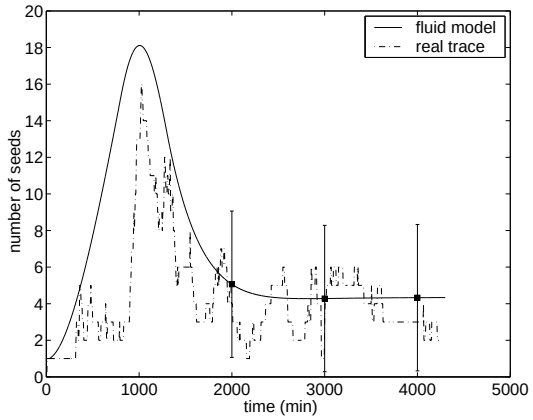

Figure 9: Experiment 3 : Evolution of the number of seeds

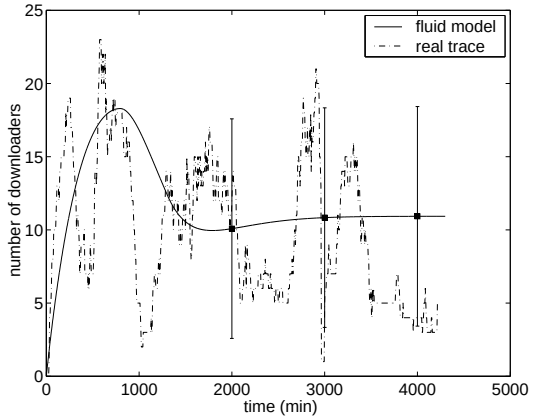

Figure 10: Experiment 3 : Evolution of the number of downloaders

file and the fluid model is calculated by using the above measured parameters. For the fluid model, we also numerically calculate the standard deviation from the steady state network parameters and plot the error bar for 95% confidence intervals. From Fig. 9, we see that the fluid model captures the evolution of the number of seeds well. In Fig. 10, the oscillation of the number of downloaders is more significant. This is because that the file is not very popular and the arrival rate λ is small. Hence, our model is only an approximation of the real network. But despite this, we can see that the oscillation is within the level suggested by the 95% confidence interval.

6 Multipath Routing

In most prior models of Internet congestion control, it has been assumed that each user is assigned a single path between its source and destination. The user then reacts to congestion on its path. However, congestion may be caused indirectly due to inefficiencies in the routing protocol itself. For example,

BGP is primarily a policy-based protocol and depending upon the policy, it can sometimes choose a low bandwidth path for a source, even when an alternate high bandwidth path is available. In this project, we consider networks where multiple paths are available for each user between its source and destination, and the user can direct its flow along these paths using source routing. The amount of flow on each path is determined by the user in response to congestion indications from the routers on the path. Currently, source routing is not supported in routers in the Internet and so we have to *overlay* the network with routers that allow source routing. Consider the scenario depicted in Figure 11, which shows a network of ISP clouds connected by peering points. In this network of ISP clouds, depending

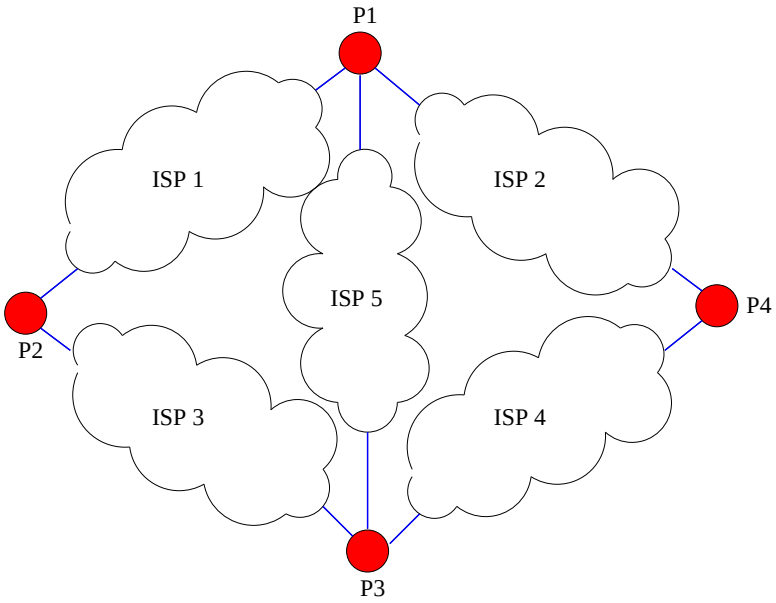

Figure 11: A network of ISP clouds. In this figure, the ISPs are connected via peering points, denoted by $P1$ through $P4$.

on the policy employed by the ISPs, a connection from ISP2 to ISP4 may be routed via peering point $P4$ even though more bandwidth may be available on a different path, say via ISP5, through peering points P1 and P3. This presents an opportunity for overlay networking to improve the service provided to the end users in the following manner: suppose that one installs overlay routers at the peering points and allows source routing at these overlay routers. Further, if the provider of the overlay routing service buys bandwidth from the ISPs, then one can create a logical network as shown in Figure 12. This would allow us to provide a service where data transfer can simultaneously take place over multiple routes in the overlay network.

Two questions immediately arise: (i) where to place these overlay routers given an existing network topology?, and (ii) given an overlay network of routers, how does one design stable congestion control algorithms that exploit the multi-path routing capability? We are interested in the second question in this project. This question was answered by Kelly, Maulloo and Tan in the case where there are no round-trip delays. In this project, we derived a stability condition when there is feedback delay in obtaining the congestion information. The key idea is that slowly traffic will be shifted from congested

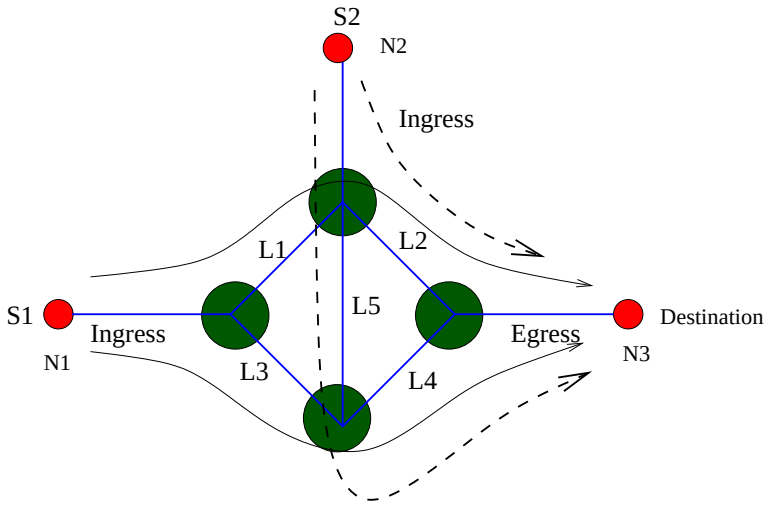

Figure 12: A logical network formed from the network in Figure 11 by overlaying routers and using virtual pipes through the ISP clouds. In this example, two sources S_1 and S_2 are transferring data to a single destination using two paths each

paths to less congested paths. If this process is carried out sufficiently slowly, then large oscillations can be avoided in the network and the network reaches a fair operating point.

7 Congestion Control in Wireless Networks

The wireless channel is a shared medium over which many users compete for resources. Since there are many users, it is important to allocate this shared resource in a fair manner among the users. Further, since the available spectrum is limited, it is also important to efficiently use the channel. However, the time-varying nature of the wireless environment, coupled with different channel conditions for different users, poses significant challenges to accomplishing these goals. Moreover, the lack of availability of channel and arrival statistics further complicates the solution.

We assume that the packets destined for the different receivers are stored in separate queues. The scheduler is responsible for allocating resources to the different queues as a function of the current channel conditions as well as the queue lengths. Prior work on this problem can be largely classified into two main categories:

- *Throughput-optimal scheduling*: Here it is assumed that the mean arrival rates of the packets into each queue lie within the capacity region (the set of sustainable arrival rates) of the channel. However, neither the actual arrival rates nor the channel capacity region is assumed to be known. The scheduler is allowed to know the current queue lengths and the current channel conditions. It has been shown that allocating resources to maximize a queue-length-weighted sum of the rates (which are feasible in the current time slot) is a stabilizing policy. Such policies are called *throughput optimal* since the queues are stable if the arrival rates lie within the capacity region.

- *Fair Scheduling:* An obvious drawback of throughput-optimal policies is that no traffic policing is enforced. For instance, if one or more sources misbehave and increase their arrival rates so that the set of arrival rates lies outside the capacity region, then the system becomes unstable. In other words, all flows will be penalized due to the behavior of a few misbehaving flows. Thus, an alternative is to provide some degree of flow isolation at least in the long term, by allocating resources in a fair manner to the various queues. It was shown in the literature that proportional fairness can be achieved in TDMA cellular networks by scheduling the user which has the largest ratio of the achievable data rate at the current instant to the average rate that it has been allocated so far.

From an applications point of view, throughput-optimal scheduling as described above is more suitable for inelastic traffic where the sources do not adapt their transmission rate based on congestion in the network. In this case, admission control is required to ensure that the arrival rates lie within the capacity region of the network and further, in the case of wireless networks, due to the time-varying nature of the network, an appropriate scheduling algorithm is required to ensure that the network can stably serve the admitted traffic. On the other hand, fair scheduling is more suited for elastic traffic sources which can adjust their traffic rates in response to feedback from the network regarding the network conditions. Without such a rate-control mechanism, fair scheduling would either lead to under utilization (when a traffic source is not generating enough data to make use of the bandwidth allocated to it) or packet losses or large delays (when a traffic source is generating data at a much larger rate than the rate allocated to it by the base station).

In this project, we are interested in allocating resources to elastic sources whose utilities are described by concave functions. Specifically, user i derives a utility $U_i(a_i)$ when it transmits at rate a_i. For ease of exposition, we consider utility functions of the form

$$U_i(a_i) = \beta_i \frac{a_i^{(1-m)}}{(1-m)},$$

where m is a positive constant and β_i is some fixed weight, which can be different for different users. Thus, we consider $m-$weighted proportionally fair resource allocation. As $m \to 1,$ this allocation converges to the weighted proportionally fair allocation and as $m \to \infty,$ it gives the weighted max-min fair allocation. We assume that congestion information is conveyed to the sources by putting the corresponding congestion price in the ACK packets. Each source react to its congestion price by choosing its transmission rates such that its marginal utility ($U_i'(a_i)$) is equal to the congestion price. We take the queue length at the base station to be the congestion price. In the Internet context, this is a special case of what is known as the dual algorithm. In wireline networks, this interpretation of queue length (or delay) as the congestion price naturally arises from an convex optimization perspective where the resource constraints are linear. However, in wireless networks, this interpretation is not immediately obvious since the resource constraints are not necessarily linear. Despite this, we show that the dual algorithm at the sources, along with queue-length-based scheduling at the base station, can be used to approximate weighted proportional fairness arbitrarily closely, where the approximation depends on the choice of a certain parameter used in the congestion control algorithm.

The algorithms developed for wireless networks in this project can be immediately implemented in cellular networks with a base station to perform the scheduling, i.e., the single-hop case. The re-

sults can also be extended to cover the multi-hop network case. However the scheduler is not implementable in the absence of a base station since then there is no central scheduler to solve the wireless resource allocation problem. The challenging problem of obtaining reasonable distributed solutions for the scheduling problem is still the hurdle whether one considers the throughput-optimal scheduling problem or the fair scheduling problem that we have considered here.

One penalty for achieving user-defined fairness (as opposed to network-dictated fairness) is the possibility of large delays at the base station buffers. We can alleviate this problem by implementing the base station scheduler using virtual queues described earlier in this report. For each flow, the base station maintains a counter called the virtual queue. As an example, consider flow i. The virtual queue of flow i keeps track of a virtual queue length, where the virtual queue length of flow i is simply the length of a queue whose arrivals are the same as that of flow i, but whose service rate is always a fixed fraction $\rho < 1$ of the actual service rate. Therefore, the size of the virtual queue will always be larger than the actual queue-length. The congestion feedback given to user i is the virtual queue length and therefore, user i will reduce its arrival rate well before its real queue builds up significantly. See Figure 13 for the model from flow i's perspective.

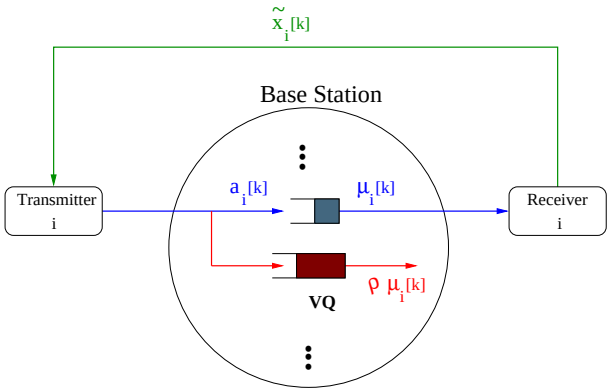

Figure 13: The virtual queue implementation at the base station.

By choosing the ρ parameter appropriately the delay levels and the packet loss probabilities can be adjusted: the lower the ρ, the lower the actual queue lengths. However, there is a possible loss in throughput by choosing $\rho < 1$. Simulations not reported here show that, by choosing ρ close to 1, but not equal to 1, we can reduce the queue lengths dramatically while maintaining close to 100% throughput.

8 Network Economics

Studying the allocation of resources to strategic agents – agents that try to optimize local objective functions that may not be the same as system-wide objectives – is the focus of micro-economics. In this project, we explored generic allocation models inspired by modern communication and computation resources, which have many characteristics not present in earlier models. Two characteristics in particular are:

1. Users/ agents may be very heterogeneous, and properly modelling their value functions may not be possible.

2. Allocation may have to be carried out under constrained communication: it may not be possible for agents to reveal much information about their preferences to the allocation mechanism.

These characteristics give rise to a few common themes to the otherwise diverse models and analysis presented in this report. The most important one, arising from the first observation about heterogeneous agents, is that all the models have a "worst-case" as their central concept: a worst case taken over very broad spectrum of scenarios under which the resource might have to be allocated. In particular, minimal assumptions are made on the "types" or value functions of the agents. This is in contrast to say, Bayesian-Nash analysis, where a-priori probability assumptions are made and the average case is considered.

Our work addresses the resource allocation problem in the economics literature under auction theory. The users send bids to the network. The network allocates the resources and charges the users following some mechanism. In the rest of this report, we always call a network user a *buyer* and frequently call the network the *seller*. The buyers have valuation functions determining the value of the resources allocated to them. Each buyer tries to maximize his value by adjusting his own bid. Therefore, the auction is formulated as a game. An allocation is *efficient* (i.e., socially optimal) if the aggregate value of the buyers is maximized. In this report, we only consider a network with a single network manager, who wishes to allocate network capacity efficiently. Hence, in the setting of auctions, there are multiple buyers and one seller. Furthermore, the valuation of the buyers are deterministic, which implies that the auction game is a Nash game. Note that there are some special properties of the auction game on a network. First, the resources such as capacity are infinitely divisible. Next, the resources in a network are inter-connected and a buyer bids for the resources along his path.

Our work consisted of two parts:

In the first part, two different auction game models are discussed. Basically, both models are extensions of the auction game on a single link network. Since a buyer in the game uses a path through the network which is combined with several links, the resources it requests from the network is the rate along the path. One extension of the auction game on single link network is to let the buyers bid for the capacity on each link separately. We call this the *Itemized bid game* since a buyer should have a bid vector along the path. Another model is the *Sum bid game*, which allows the buyers to have single value bids. The network allocates the rate along the path by a weighted proportionally fair fashion. We discuss and compare the condition for a Nash equilibrium point in these two games on a two-link network. We argue that the sum bid game captures more fully the interactions among strategic buyers. However, the sum bid game has worse efficiency compared to the itemized bid game and the payoff function of a buyer in the sum bid game need not to be concave.

In the second part, we focus on the auction game on a single link network and design an efficient mechanism. That is, we design an efficient mechanism for allocation of an infinitely divisible good. We present an efficient mechanism in which the bids of the buyers are scale-valued. Basically, the efficient allocation can be achieved by allocating the good in proportion to the bids and charging the buyers some non-uniform prices. A buyer has incentive to bid for efficiency by the rule that he has to pay an amount equal to the externality he exerts on other competing buyers. Also, the mechanism is implemented in a decentralized dynamic system. It allows the buyers to update their bids unilaterally

in seeking personal payoff maximization and finally the system reaches the efficient allocation.

9 Conclusions

In this project, we have demonstrated that simple mathematical models can accurately capture the dynamics of a complex interconnected system such as the Internet and wireless networks. We have developed models at various time scales of interest and showed that these models can be used to both predict network performance as well as to design new protocols to improve the performance. The majority of the models developed are in the form of deterministic delay differential equations. We have also used stochastic models to justify that, under practical network operating conditions, the stochastic models can be well approximated by the delay differential equation models. The models developed in this project provide a clear insight into the operation of complex communication networks and also allow us to improve the design of such networks.